代码示例

计数排序的核心代码如下。

```
int a[N + 10],b[N + 10],c[W + 10]; //W 为值域大小
for (int i = 1; i <= n; i++) c[a[i]]++;
for (int i = 1; i <= W; i++) c[i] += c[i - 1];
for (int i = 1; i <= n; i++) b[c[a[i]]--] = a[i];
```

参考词条

排序的基本概念

延伸阅读

THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2013：108-110.

(叶金毅)

1.4.6 搜索算法

搜索又称状态空间搜索，是指通过探索从初始状态到目标状态的路径以完成问题求解的算法。

典型的搜索算法有深度优先搜索、广度优先搜索(宽度优先搜索)、启发式搜索等。

搜索算法可以看成在以状态为结点、根据状态间的扩展关系确定连边的图上进行遍历的算法，可以将搜索中的一个状态称为搜索中的一个结点，起始状态和目标状态分别称为起始结点和目标结点。

在搜索一个连通图 $G=(V,E)$ 时，若将每个结点第一次被生成时的边的集合记为 E'，则 $G'=(V,E')$ 是图 G 的一棵生成树。

1.4.6.1 深度优先搜索

深度优先搜索在算法执行的过程中每次将目前未扩展的状态中最深的状态进行扩展，生成新的状态。一般使用递归调用的方式实现深度优先搜索算法，有时也会通过模拟栈以非递归的方式实现。

使用递归调用的方式实现深度优先搜索算法，需要实现一个递归函数，可命名为 dfs(p)，参数 p 包含状态信息，dfs 的基本步骤如下。

(1) 如果状态 p 已经被访问过，则退出函数；否则将 p 标记为访问过。

(2) 如果 p 是目标状态，则找到解，退出函数。

(3) 将 p 进行扩展，生成出子状态，对于每一个子状态 q，调用函数 dfs(q)，如果有任意一次调用找到解，则退出函数。

深度优先搜索对应图中的深度优先遍历，通常也将图中的深度优先遍历称为深度优先搜索。按照深度优先搜索生成的子图称为深度优先搜索生成树，简称深搜生成树。

在深度优先搜索中，可以仅计算出当前正在展开或访问的一系列状态，并用栈进行记录。算法的空间复杂度为 $O(d)$，其中 d 为最大搜索深度。算法的时间复杂度为 $O(n)$，其中 n 为全部状态的总数。

参考词条

深度优先遍历

典型题目

1. NOIP2000 提高组 单词接龙
2. USACO Training Checker Challenge
3. USACO2005Dec Scales

（胡伟栋）

1.4.6.2 广度优先搜索

广度优先搜索又称宽度优先搜索，在算法执行的过程中每次将目前最浅的未扩展状态进行扩展，生成新的状态。一般使用一个队列来实现广度优先搜索算法。广度优先搜索算法的基本步骤如下：

（1）定义队列 Q 用于保存已生成但未扩展的状态，初始仅包含初始状态；

（2）定义集合 S 用于判重，初始仅包含初始状态；

（3）从 Q 中取出队首元素 p，对 p 进行扩展，如果扩展中生成的状态不在 S 中，则加入 S，并加入 Q；

（4）若在步骤（3）生成了目标状态，则找到了解，否则重复执行步骤（3）直到 Q 为空。

广度优先搜索对应图中的广度优先遍历，通常也将图中的广度优先遍历称为广度优先搜索。按照广度优先搜索生成的子图称为广度优先搜索生成树，简称广搜生成树。

在广度优先搜索中，需要记录当前已访问的状态信息。记 b 为每个状态的子状态数量，d 是最大搜索深度，则算法的时间和空间复杂度均为 $O(b^d)$。

参考词条

广度优先遍历

典型题目

1. NOIP2002 提高组 字串变换
2. NOIP2017 普及组 棋盘
3. CSP2019-J 加工零件

1.4.7 图论算法

1.4.7.1 深度优先遍历

从图中某一顶点出发，访问图中其余顶点，且每一个顶点仅被访问一次，这种访问的过程叫作图的遍历。常见的图的遍历方式有深度优先遍历和广度优先遍历。图的深度优先遍历也称为深度优先搜索(Depth-First Search，DFS)。

从图中的某个顶点 u 出发，访问此顶点后，依次从 u 的未被访问的邻接点出发深度优先遍历图，直到图中所有和 u 有路径相通的顶点都被访问到；若此时图中尚有顶点未被访问，则另选一个未曾访问的顶点作为起始点重复上述过程，直至图中的所有顶点都被访问到为止。深度优先遍历算法常用递归来实现，使用栈结构来存放遍历的顶点。算法主要步骤如下。

(1) 从某个顶点 u 出发，并标记 u 为已访问状态，表示 u 点已被访问。

(2) 访问 u 的邻接点 v，若 v 点未被访问，则从 v 点出发继续深度优先遍历。

在对图进行深度优先遍历时，按照顶点首次被访问的顺序，对各顶点进行编号，所得到的编号序列被称为 DFS 序，常记作 dfn。

在图 1.9 中，从顶点 A 出发对该无向图进行深度优先遍历，可得到一组遍历序列 A、C、D、F、G、E、B。图中的数字即为各顶点的 DFS 序编号。

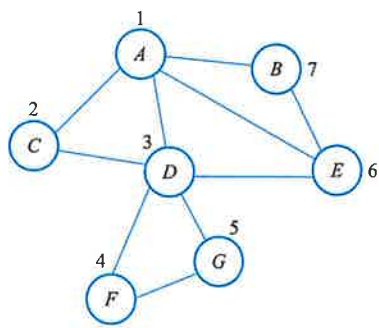

图 1.9 图的深度优先遍历

代码示例

深度优先遍历的核心代码如下。

```
void dfs(int u)
{
    cout << u << " ";
    vis[u] = true; //标记顶点 u
    for (int i = head[u]; i; i = e[i].next)
    { //枚举所有与 u 相邻的顶点
        int v = e[i].v;
        if (vis[v])
            continue; //若该顶点被访问过,则跳过
        dfs(v); //从 v 顶点出发,继续深度优先遍历
    }
}
```

参考词条

1. 递归函数
2. 图的定义与相关概念

延伸阅读

THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2013：349-354.

(李绍鸿　谢秋锋)

1.4.7.2　广度优先遍历

图的广度优先遍历也称为宽度优先遍历或广度优先搜索(Breadth-First Search，BFS)。

从图中某一顶点 s 出发，依次访问 s 的各个未曾访问过的邻接点，并借助队列将这些点保存起来，再分别从这些邻接点出发并依次访问它们未曾访问过的邻接点，并将新访问到的邻接点依次保存在队列中。这个过程使得“先被访问的顶点的邻接点”先于“后被访问的顶点的邻接点”被访问，直到图中所有与顶点 s 有通路的顶点都被访问完毕。若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为起始点重复上述过程，直至图中所有顶点均被访问到为止。

因此，广度优先遍历的过程是以 s 为起点，由近至远，依次访问和 s 有路径相通且路径长度为 1、2、…的顶点的过程。算法的主要步骤如下。

(1) 起点 s 入队，并标记 s 为已访问状态。

(2) 若队列非空，则重复以下步骤：队首顶点 u 出队，访问 u 的所有邻接点 v，若 v 未被访问，则让 v 入队并标记 v 为已访问状态。

在图 1.10 中，从顶点 A 出发对该无向图进行广度优先遍历，可得到一组遍历序列 A、C、D、E、B、F、G。图中的数字即对应各顶点被访问的次序。

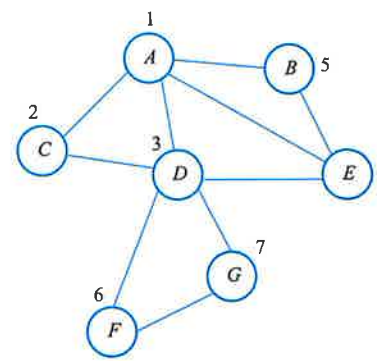

图 1.10　图的广度优先遍历

代码示例

广度优先遍历的核心代码如下。

```
vis[s] = true; //标记起始点 s 为访问过
queue<int> q;
q.push(s); //将访问过的顶点存入队列中
while (!q.empty())
{                          //当队列不为空,则扩展队头元素
    int u = q.front(); //取出队头顶点 u
```

```
        cout << u << " ";
        q.pop();
        for (int i = head[u]; i; i = e[i].next)
        { //枚举所有与队头顶点 u 相邻的顶点
            int v = e[i].v;
            if (vis[v])
                continue; //如果该顶点被访问过,则跳过
            vis[v] = true; //否则标记 v 为访问过
            q.push(v); //将顶点 v 存入队列中
        }
    }
```

参考词条

1. 队列
2. 图的定义与相关概念

延伸阅读

THOMAS H C，CHARLES E L，RONALD L R，et al. 算法导论(原书第 3 版)[M]. 殷建平，徐云，王刚，等译. 北京：机械工业出版社，2006：343-348.

(李绍鸿　谢秋锋)

1.4.7.3　泛洪算法

洪水填充算法(flood fill algorithm)，也称为泛洪算法，用于将格点图某一块连通区域内的所有格点状态修改为目标状态。状态通常用颜色表示。一般处理方法是，从一个起始顶点开始把附近与其连通的顶点填充成新的颜色，直到连通区域内的所有顶点都被处理过为止。因为其思路类似洪水从一个区域扩散到所有能到达的其他区域而得名。

洪水填充算法可以使用深度优先遍历、广度优先遍历或其他方式实现。

洪水填充算法最常见的有四邻域填充法和八邻域填充法。四邻域包括上、下、左、右四个方向，而八邻域包括四邻域的四个方向和对角线四个方向，如图 1.11 所示。

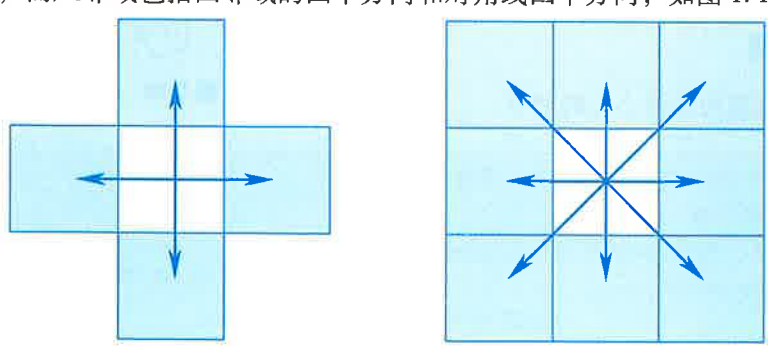

图 1.11　四邻域与八邻域